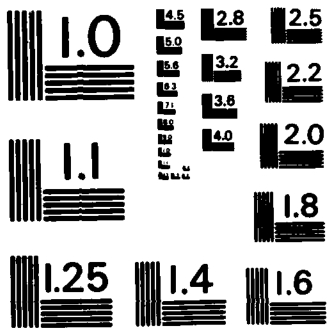


MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

15

Computing the Ambiguity Surface

R. Tolimieri* and S. Winograd

Mathematical Sciences Department, IBM Thomas J. Watson Research Center,
Yorktown Heights, New York 10598

* The first author received O.N.R. support during the period of this work.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER [illegible]
2. GOVT ACCESSION NO. A121034
3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) COMPUTING THE AMBIGUITY SURFACE
5. TYPE OF REPORT & PERIOD COVERED
6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) R. TOLIMIERI, S. WINOGRAD
8. CONTRACT OR GRANT NUMBER(s) 7300/N000 14-82-C-0230
9. PERFORMING ORGANIZATION NAME AND ADDRESS CUNY NYC
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS CUNY NYC
12. REPORT DATE OCT 13, 1982
13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)
15. SECURITY CLASS. (of this report)
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release: distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)
18. SUPPLEMENTARY NOTES NONE

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

DTIC COPY INSPECTED 2

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
RADAR, SONAR
AMBIGUITY SURFACE
FAST FOURIER TRANSFORM
DECIMATING FILTER
POLYNOMIAL APPROXIMATION

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
In this work computational methods for evaluating the digital ambiguity surface are studied and compared.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Introduction

The ambiguity function or surfaces provides the designer of modern radar and sonar technology a measure of the effectiveness of a given matched-filter signal in determining the simultaneous evaluation of range and velocity of a moving object to the particular conditions of target environment. In this work, we will be concerned with efficient computational methods for evaluating this function by computers.

The ambiguity function is defined, for our purposes, as the absolute value of the function

$$G(j, n) = \sum_{k=0}^{K-1} x_k y_{k+j}^{*} \exp\left(-2\pi i \frac{nk}{K}\right)$$

where J, K, N are positive integers, $0 \le k < K$, $0 \le j \le J$, $-N \le n \le N$ and x_k, y_{k+j} are data. We will denote $\sqrt{-1}$ by i.

In section 2., we will view the computation of G(j,n) as a filter or convolution (of $x_k \exp(-2\pi i \frac{nk}{K})$ by y_k^{*}), while in section 3., G(j,n) is interpreted as a Fourier transform (of $x_k y_{k+1}^{*}$). Several departures, from usual methods, making use of special circumstances, will be described, significantly increasing the computational efficiency.

In section 3., as well, the technique of passing a long sequence through an F.I.R. filter with decimation and computing the discrete Fourier transform (D.F.T.) of the shorter decimated sequence will be applied to the problem. We will follow the approach of [1].

In section 4., a method based on polynomial approximation theory will be introduced and compared to the decimating filter technique. Indeed, after some initial differences, the final stages of computing by the approximation technique coincides with those of the decimating filter method.

2. Filter Method

Let $A(j,n) = | G(j,n) |$. The first method for computing $A(j,n)$ is based upon writing $G(j,n)$ as a filter and applying fast Fourier transform (FFT) techniques to the resulting computation. Let

$$z_k(n) = x_k \exp\left(- 2\pi i \frac{nk}{K}\right).$$

For each n, $-N \le n \le N$, we can write

$$G(j,n) = \sum_{k=0}^{K-1} z_k(n)\, y^*_{k+j} \qquad , 0 \le j \le J \tag{2.1}$$

which expresses $G(j,n)$ as the first $J+1$ outputs of a K-tap filter. (That is, we view the sequence $\{y^*_\ell\}$ as the data and the sequence $\{z_k(n)\}$ as the tap values.)

Equivalently, if we set

$$\tilde{z}_k(n) = \begin{cases} z_k(n) & , 0 \le k < K \\ 0 & , K \le k < K + J \end{cases} \qquad \text{(padding)}$$

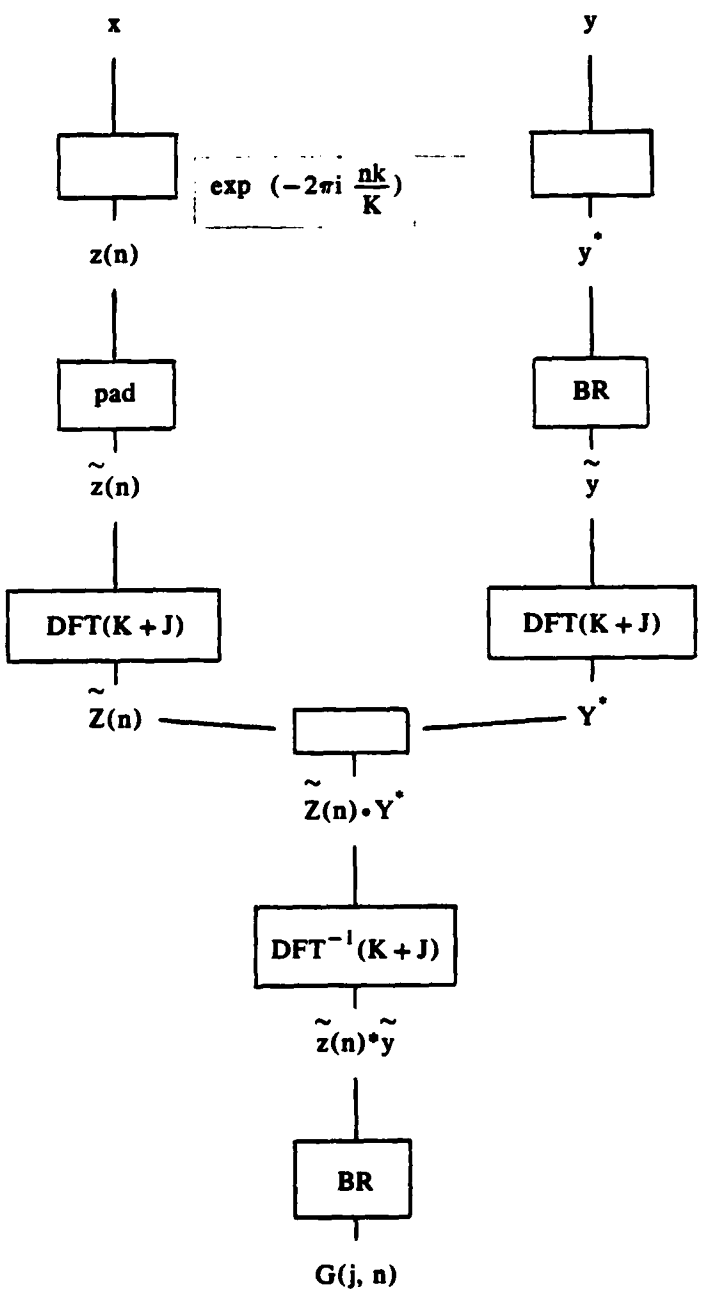
$$\tilde{y}_\ell = y^*_{K+J-\ell} \qquad , 0 \le \ell < K + J \qquad \text{(bit-reversal)}$$

then for each n, $-N \le n \le N$, we can write $G(j,n)$ as the cyclic convolution.

$$G(j,n) = (\tilde{z}(n) * \tilde{y})_{K+J-j} \qquad , 0 \le j \le J. \tag{2.2}$$

Computing the cyclic convolution by standard DFT techniques yields Flow chart A for the computing of $G(j,n)$, $0 \le j \le J$. In general, the DFT of a sequence x will be denoted by X.

Flow Chart A


x
y
$\exp(-2\pi i \frac{nk}{K})$
$z(n)$
y^*
pad
BR
$\tilde{z}(n)$
$\tilde{y}$
DFT(K + J)
DFT(K + J)
$\tilde{Z}(n)$
Y^*
$\tilde{Z}(n) \cdot Y^*$
$DFT^{-1}(K+J)$
$\tilde{z}(n) * \tilde{y}$
BR
G(j, n)

Denote by FFT(K) the number of complex multiplications used by the FFT methods to compute the DFT(K). Note $\tilde{y}$ is independent of n and Y^* need be computed only once in Flow chart A. It follows that the number of complex multiplications used by Flow chart A to compute $G(j,n)$, $0 \le j \le J$, $-N \le n \le N$, is given by

$$(2.3) \qquad (2N + 1)\,(K + 2\,\mathrm{FFT}(K + J) + (K + J)) \; + \; \mathrm{FFT}(K + J).$$

The number given by (2.3) will be denoted by $\pi_1 = \pi_1(N, K, J)$. Table I describes the numerology for three choices of N, K and J. In the examples, we will take $\mathrm{FFT}(K) = \frac{1}{2} K \log_2 K$.

Table I

K	J	N	Π_1
2^{13}	2^{13}	50	$3145 \cdot 2^{13}$
$2^{14}-700$	700	50	$1623 \cdot 2^{14}-70{,}700 \sim 1619 \cdot 2^{14}$
$2^{14}-70$	70	50	$1623 \cdot 2^{14}-7{,}070 \sim 1623 \cdot 2^{14}$

In certain cases, we can improve on the computational efficiency of the previous method. Suppose, for instance, we have $K = J$. The computation of $\tilde{Z}(n)$, $-N \le n \le N$, can be made as follows.

Pad the sequence x by setting

$$\tilde{x}_k = \begin{cases} x_k & , 0 \le k < K \\ 0 & , K \le k < 2K \end{cases}$$

and denote the DFT(2K) of $\tilde{x}$ by $\tilde{X}$. Then,

$$\begin{aligned}\tilde{X}_{u+2n} &= \sum_{k=0}^{K-1} x_k \exp\left(-2\pi i \frac{u+2n}{2K} k\right) \\ &= \sum_{k=0}^{K-1} z_k(n) \exp\left(-2\pi i \frac{uk}{2K}\right) \\ &= \sum_{k=0}^{2K-1} \tilde{z}_k(n) \exp\left(-2\pi i \frac{uk}{2K}\right).\end{aligned}$$

It follows that

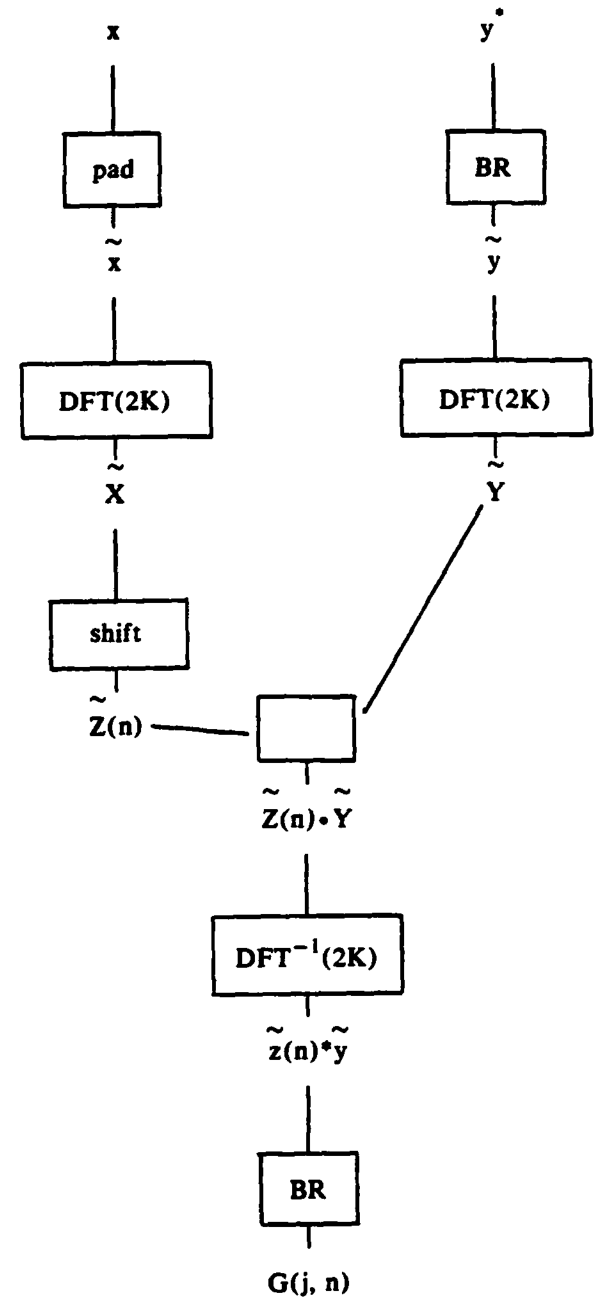
$$(\tilde{Z}(n))_u = \tilde{X}_{u+2n} \quad , 0 \le u < 2K, \ -N \le n \le N. \tag{2.4}$$

Once $\tilde{X}$ has been computed, without any additional complex multiplications, we can compute $\tilde{Z}(n)$, $-N \le n \le N$. Specifically, $\tilde{Z}(n)$ is given by cyclically shifted, mod 2K, the sequence $\tilde{X}$ 2n places.

Flow chart B descibes how the observation applies to the computation of G(j,n). Also, the number of complex multiplications used on Flow chart B is given by

$$(2N + 1)(2K + \text{FFT}(2K)) + 2\ \text{FFT}(2K). \tag{2.5}$$

<u>Flow Chart B</u>



x
y^*
pad
BR
$\tilde{x}$
$\tilde{y}$
DFT(2K)
DFT(2K)
$\tilde{X}$
$\tilde{Y}$
shift
$\tilde{Z}(n)$
$\tilde{Z}(n) \cdot \tilde{Y}$
$DFT^{-1}(2K)$
$\tilde{z}(n) * \tilde{y}$
BR
$G(j, n)$

__Example__. For $K = J = 2^{13}$ and $N = 50$, substituting into (2.5) yields 1644.2^{13} which is a 50% savings as compared to π_1 as given in Table I.

Essentially, the same argument holds if J is any integer multiple of K. The case when K ia a multiple of J will now be considered. Suppose, for instance, $K = 2J$. Pad x by setting

$$\tilde{x}_k = \begin{cases} x_k & , 0 \leq k < K \\ 0 & , K \leq k < \frac{3}{2}K \end{cases}$$

and let $\tilde{X}$ be the DFT($\frac{3}{2}$ K) of $\tilde{x}$, as above,

(2.6) $$(\tilde{Z}(2n))_u = \tilde{X}_{u+3n} \qquad , 0 \leq u < \frac{3}{2} K, \ -N \leq 2n \leq N.$$

it follows that $\tilde{Z}(2n)$ can be found by cyclically shifting, mod ($\frac{3}{2}$ K), the sequence $\tilde{X}$ 3n places. Analogously, if we set

$$\tilde{x}_k(1) = \begin{cases} x_k \exp(-2\pi i \frac{k}{K}) & , 0 \leq k < K \\ 0 & , K \leq k < \frac{3}{2}K \end{cases}$$

then

(2.7) $$(\tilde{Z}(2n+1))_u = \tilde{X}(\bar{1})_{u+3n}$$

Thus, the number of complex multiplications used is

(2.8) $$(2N+1)(\mathrm{FFT}(\tfrac{3}{2} K) + \tfrac{3}{2} K) + 3\ \mathrm{FFT}(\tfrac{3}{2} K) + K.$$

__Example 2__. Suppose $K = 2^{13}$, $J = 2^{12}$ and $N = 50$. Flow chart A uses approximately $2^{12} . 6000$ complex multiplications. By (2.7), since $K = 2J$, we can compute the case in ar ximately $2^{12} . 4650$ complex multiplications.

3. Transform Method

For each j, $0 \leq j \leq J$, consider the sequences, defined mod K,

$$\left.\begin{aligned} y_k(j) &= y_{k+j} \\ u_k(j) &= x_k y_k^*(j). \end{aligned}\right\} \quad , 0 \leq k < K,$$

We can write

$$G(j, n) = U_n(j) \qquad , -N \leq n \leq N, \tag{3.1}$$

where U(j) is the DFT(K) of u(j). Flow chart C outlines the computation of G(j,n) by this method. The number of complex multiplications used by the method is

$$(J + 1)(\mathrm{FFT}(K) + K). \tag{3.2}$$

We will denote the number given by (3.2) by $\pi_2 = \pi_2(K, J)$. Table II reconsiders those examples of Table I by this new method and compares these values corresponding values π_1 given in Table I.

Flow Chart C

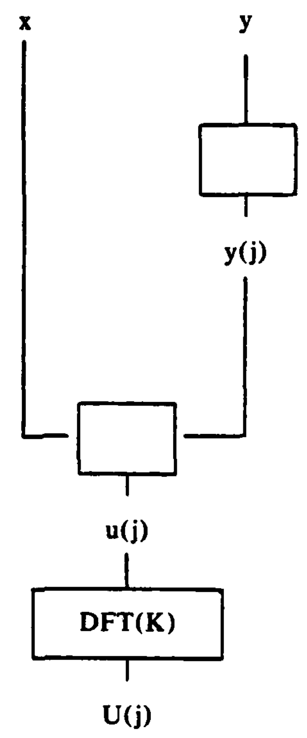


Table II

K	J	N	π_2	π_1
2^{13}	2^{13}	50	$2^{12} . 15 . 8193$	$\sim \frac{1}{40} \pi_2$
$2^{14}-700$	700	50	$< 5586 . 2^{14}$	$\sim \frac{2}{7} \pi_2$
$2^{14}-70$	70	50	$< 567 . 2^{14}$	$\sim 3 \pi_2$

Note that in the last case the transform method is more computationally efficient than the filter method.

In the most important applications, N<<K and we will assume this in all that follows. As evidenced by (3.2), the method of Flow chart C does not take into account the relative size of N as compared to K. Generally speaking, the problem is to compute the DFT(K) of

the long sequence u(j) for only a small interval of frequencies $-N \leq n \leq N$. We will follow the approach of [1]. As we will see, the long sequence u(j) will be passed into an FIR filter with decimation and the DFT of the shorter decimated sequence will be computed. The amount of complex multiplications used will depend upon the number of taps of the filter and the decimation rate. However, the DFT of the decimated sequence yields only an approximation to G(j,n), due to aliasing errors.

Consider to begin with, a K-tap filter given by

$$h = (h(0), h(1), \ldots, h(K-1))$$

satisfying the following properties

(I) $h(k) = h(-k)$.

(II) h(k) vanishes outside an interval, $-M \leq k \leq M$, for some integer M.

(III) Let H be the DFT(K) of h. Then $H(0) = 1$ and H(n) never vanishes, $-N \leq n \leq N$.

Passing the sequences u(j) through the filter h we determine the sequence v(j) given by

$$v_\ell(j) = \sum_{k=0}^{K-1} u_{\ell-k}(j)\, h(k) \qquad , 0 \leq \ell < K,$$

or equivalently $v(j) = u(j) * h$. Let U(j), V(j) denote the DFT(K) of u(j), v(j), respectively. Then,

$$V(j) = U(j) \times H.$$

Suppose $K = R \times L$, R, L integers and assume $N < R$. Consider the (L:1) decimation of v(j) given by

$$b_r(j) = v_{rL}(j) \qquad 0 \le r < R$$

and denote the DFT(R) of b(j) by B(j). A standard computation, see [1], shows that

$$B_r(j) = L^{-1}(V_r(j) + E_r(j)) \qquad , 0 \le r < R,$$

where

$$E_r(j) = \sum_{\ell=1}^{L-1} U_{r+\ell R}(j)\ H(r + \ell R) \qquad , 0 \le r < R.$$

Condition (III) implies we can write

$$U_n(j) = H(n)^{-1}\, L\, B_n(j) - H(n)^{-1}\, E_n(j) \qquad , -N \le n \le N.$$

We plan to approximate $G(j,n) = U_n(j)$ by $H(n)^{-1} L\, B_n(j)$. The error

$$H(n)^{-1} E_n(j) = \sum_{\ell=1}^{L-1} U_{n+\ell R}(j)\ H(n)^{-1}\, H(n + \ell R)$$

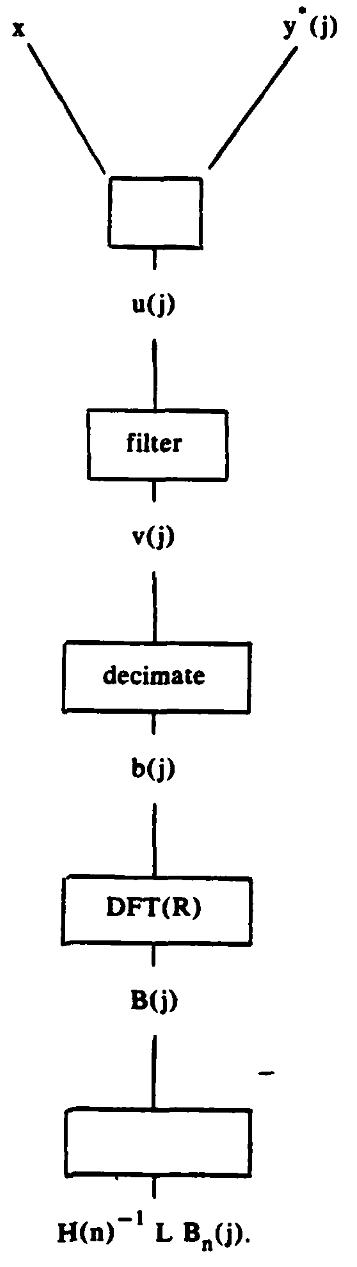
is due to the aliasing of $U_{n+\ell R}(j)$ into the frequencies $-N \le n \le N$ attenuated by the factors $H(n)^{-1}\, H(n + \ell R)$, $-N \le n \le N$, $0 < \ell < L$. An analysis of this aliasing error was carried out in [1]. For our purposes, it suffices to remark that the filter h should be chosen to minimize the factors $H(n)^{-1}\, H(n + \ell R)$, for $-N \le n \le N$, $0 < \ell < L$, in the sense of the sup norm. Condition (I) permits the application of the Remez algorithm as contained in [1] to achieve the minimization of these factors subject to conditions (II) and (III). At the same time, the Remez algorithm produces the uniform bound of these factors.

For each M and L, the Remez algorithm determines a filter h satisfying (I), (II) and (III) and minimizing the factors $H(n)^{-1}\, H(n + \ell R)$ over the aliasing frequencies. For fixed L, the error term increases as the number of taps, $2M + 1$, of h decreases. Also, increasing the decimation rate L, with fixed M, increases the error term. Thus, M large and L small produces small error terms. However, as we will soon see, the computation of $H(n)^{-1}L\ B_n(j)$ is most computationally efficient when M is small and L is large.

Variations of these ideas are considered in [1] where weighting factors over the decreasing frequencies are introduced into the Remez subroutine as a way of accounting for the a priori knowledge that certain aliasing frequencies require less attenuation than others.

Flow chart D describes the computation of $H(n)^{-1}L\ B_n(j)$, in a straightforward manner. A more computationally efficient method will eventually be given.

Flow Chart D For $0 \leq j \leq J$,



The number of complex multiplications used by Flow chart D to compute $H(n)^{-1} L B_n(j)$, $-N \leq n \leq N$, $0 \leq j \leq 2J$ is given by

(3.3) $$(J + 1)(K + 2M + 1 + FFT(R) + 2N + 1)$$

The major portion of the computation occurs in passing from x and y(j) to B(j). We will study this process more closely. First, we can write

$$\begin{aligned} b_r(j) &= \sum_{k=0}^{K-1} u_{rL-k}(j)\, h(k) \\ &= \sum_{\ell=0}^{K-1} x_{rL-k}\, h(\ell)\, y^*_{rL-k}(j) \\ &= \sum_{m=-M}^{M} x_{rL-m} h(m)\, y^*_{rL-m}(j) \end{aligned}$$

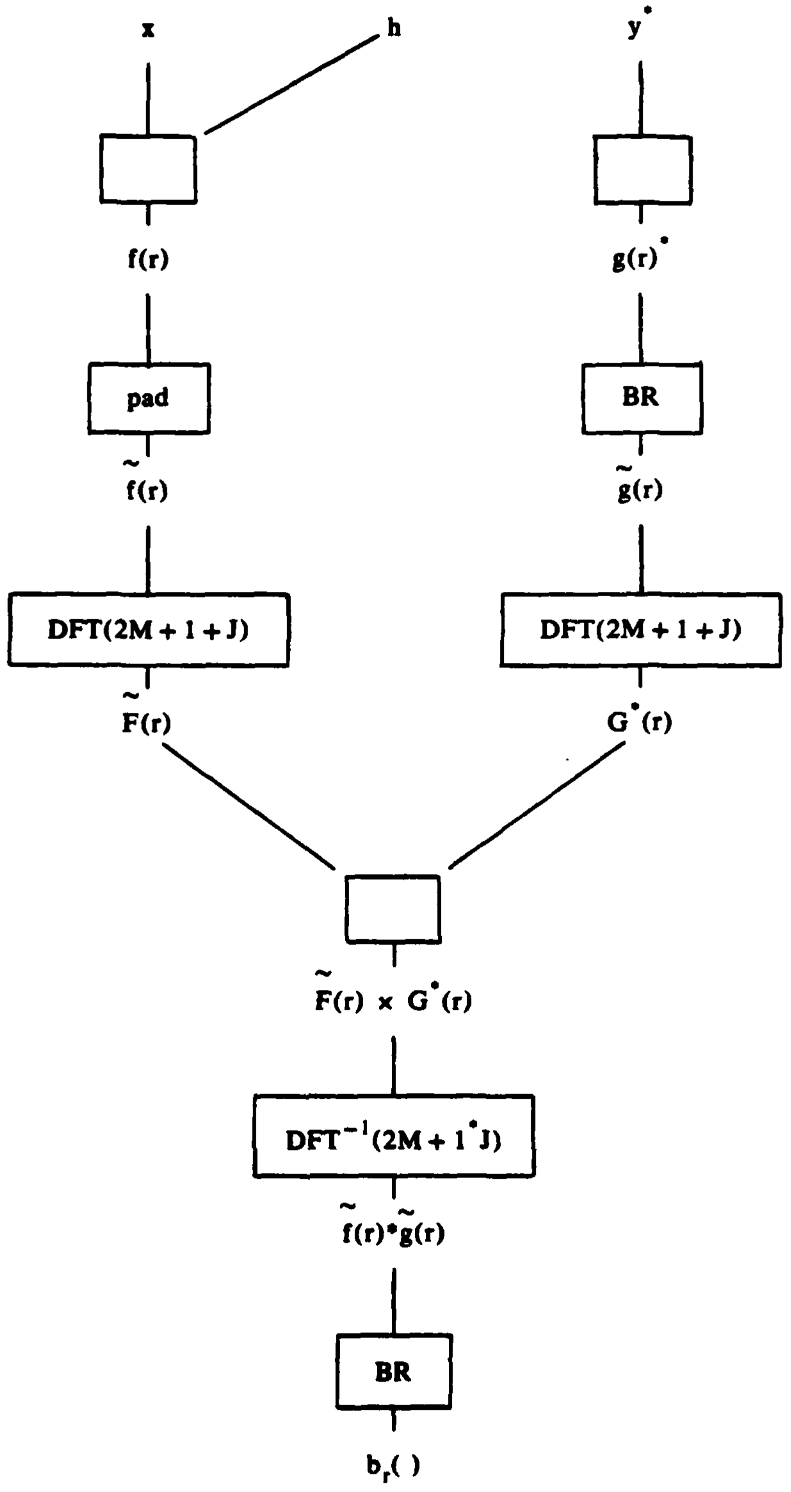
Set $f_m(r) = x_{rL+M-m} h(-M+m)$, $0 \le m \le 2M$ and $g_m(r) = y^*_{rL-M+m}$, $0 \le m \le 2M + J$. Then,

$$b_r(j) = \sum_{m=0}^{2M} f_m(r)\, g_{m+j}(r). \tag{3.4}$$

The process described in Flow chart D assumes j, $0 \le j \le J$ is fixed. We will now adopt the point of view that r, $0 \le r < R$, is fixed and use (3.4) to compute $b_r(\ )$, the sequence in j, $0 \le j \le J$, corresponding to the fixed r. Thus, (3.4), expresses $b_r(j)$ as the first (J+1)-outputs of a (2M+1)-tap filter and the methods of section 2., especially Flow chart A, can be employed. We are also assuming M is sufficiently large to make the approach computationally efficient.

Flow chart E, below, describes how for fixed r, $0 \le r < R$, the method of Flow chart A proceeds from original data to the sequence $b_r(\ )$. If Flow chart E is carried out for each r, $0 \le r < R$, then the end of Flow chart D can be applied to b(j) to compute $H(n)^{-1} L\, B_n(j)$.

<u>Flow Chart E</u> For $0 \leq r < R$,



Flow chart E uses

$$(3.5) \qquad R(2M + 1 + 3\ FFT(2M + 1 + J) + 2M + 1 + J)$$

complex multiplications to compute $b_r(j), 0 \leq j \leq J, 0 \leq r < R$, and hence $H(n)^{-1} L\ B_n(j), -N \leq n \leq N, 0 \leq j \leq J$, can be computed by

$$(3.5) \quad R(2M + 1 + 3\ FFT(2M + 1 + J) + 2M + 1 + J) + (J + 1)(FFT(R) + 2N + 1)$$

complex multiplications.

In Table III below, we will denote the number given by (3.6) by $\pi_3 = \pi_3(N, K, J, M, L)$.

<u>Table 3</u> $N = 50$

K	J	M	L	π_3	π_2
2^{13}	2^{13}	2^9-1	2^7	$> 2^{12} \cdot 508$	$2^{12} \cdot 15 \cdot 8193$

For a more complete analysis of the relationship between given values of K, J, M, L, N and the corresponding values of π_3 see Table I, pages 216 of [1].

4. <u>Approximation Theory</u>

In this section, we will obtain a method analogous to that discussed in section 3, by using ideas from approximation theory. As in section 3, the DFT(R) will be applied to the (L:1) decimation of a sequence obtained by passing K-point data through a filter. The filter will be determined by the degree of the approximation taken, in a sense explained below. Once the computation to be made is written in this form, the methods of section 3 can be employed.

Consider, again, the expression for G(j,n) given by (3.1); namely,

$$G(j,n) = \sum_{k=0}^{K-1} u_k(j) \exp(-2\pi i \frac{k.n}{K}) \tag{4.1}$$

Suppose we write $K = R \cdot L$ and $k = \ell + rL$, $0 \leq r < R$, $0 \leq \ell < L$. Substituting $\ell + rL$ for k in (4.1), we get

$$G(j,n) = \sum_{r=0}^{R-1} \exp(-2\pi i \frac{nr}{R}) \sum_{\ell=0}^{L-1} u_{rL+\ell}(j) \exp(-2\pi i \frac{n\ell}{K}). \tag{4.2}$$

The plan is to approximate $\exp(-2\pi i \frac{n\ell}{K})$ by an expression which when placed into (4.2) allows us to write the right-hand side of (4.2) as the DFT(K) of some sequence in r, $0 \leq r < R$. This DFT(R) will be taken as an approximation to G(j,n).

We will assume throughout that $N << K$. Suppose we approximate $\exp(-2\pi i \frac{n\ell}{K})$, $-N \leq n \leq N$, $0 \leq \ell < L$, by 1. The resulting error is small when N and L are small. Replacing $\exp(-2\pi i \frac{n\ell}{K})$ by 1 in (4.2), yields

$$C_n(j) = \sum_{r=0}^{R-1} c_r(j) \exp(-2\pi i \frac{nr}{R}) \tag{4.3}$$

where

$$c_r(j) = \sum_{\ell=0}^{L-1} u_{rL+\ell}(j). \tag{4.4}$$

It is easy to see that the number of complex multiplications used is

$$(J + 1)(K + \text{FFT}(R)). \tag{4.5}$$

Clearly, (4.4), expresses $c_r(j)$ as the (L:1) decimation of the filtered sequence

$$w_k(j) = \sum_{\ell=0}^{L-1} u_{k+\ell}(j) \quad , 0 \leq k < K.$$

For n, $-N \leq n \leq N$, consider that part of the unit circle given by $f(t) = \exp(-2\pi i\frac{nt}{K}$, $0 \leq t \leq L$. Denote by $q_1(t)$ the polynomial of degree one in t uniquely determined by the conditions

$$q_1(0) = f(0)$$

$$q_i(L) = f(L).$$

Thus, $q_1(t)$ is geometrically the line joining the point 1 to the point $\exp(-2\pi i \frac{n}{R})$. We can write

$$q_1(t) = (1-\frac{t}{L}) + \frac{t}{L} e^{-2\pi i \frac{n}{R}}.$$

Choose a real number $d_1(n)$ such that

$$p_1(t) = d_1(n)\, q_1(t)$$

is the best-approximation to f(t), in the sup-norm sense, over the interval $0 \leq t \leq L$, within the space of all real multiples of $g_1(t)$.

Replacing $\exp(-2\pi i\frac{n\ell}{K})$ by $p_1(\ell)$ in (4.2) we obtain

(4.6) $$d_1(n) \;\; C_n(j)$$

where

(4.7) $$c_r(j) = \sum_{\ell=0}^{L-1} \left(u_{rL+\ell}(j)\left(1-\frac{\ell}{L}\right) + u_{(r-1)L+\ell}(j)\frac{\ell}{L}\right)$$

and C(j) is the DFT(R) of c(j).

To see the analogy of this approach to the method of section 3, we proceed as follows. Let h(k), $0 \le k < K$, be defined by

$$h(k) = \begin{cases} \frac{k}{L} & , 0 \le k < L \\ L - \frac{k}{L} & , L \le k < 2L \end{cases}$$

and zero otherwise. Consider the 2L-tap filter determined by h,

$$w_t(j) = \sum_{\ell=0}^{2L-1} h(\ell)\, u_{t-L+\ell}(j) \tag{4.8}$$

By (4.7), we can write c(j) as the (L:1) decimating filter

$$c_r(j) = w_{rL}(j) \quad , 0 \le r < R. \tag{4.9}$$

Again, we are led to the computing of the DFT(R) of an (L:1) decimating filter. Here, the filter is determined by the approximation taken. Since the interpolating polynomial has degree one, we call the method, the first degree approximation method.

The techniques of section 3, can be applied to the computation of $c_r(j)$. Setting

$$x_\ell(r) = x_{(r-1)L+\ell}$$

$$x'_\ell(r) = x_\ell(r)\, h(\ell)$$

$$y'_{j+\ell}(r) = y^*_\ell(r)\, y'_{\ell+j}(r),$$

which for each r, $0 \le r < R$, expresses $c_r(\ )$ as the first (J+1)-outputs of a 2L-tap filter. The number of complex multiplications used by this method to compute

$d_1(n)\ C_n(j)$, $-N \leq n \leq N$, $0 \leq j \leq J$, is

(4.11) $\quad 2K + R(3FFT(2L + J) + 2L + J) + (J + 1)(FFT(R) + 2N + 1).$

Choose $M>0$ and consider that part of the unit circle given by $f(t) = \exp(-2\pi i \frac{nt}{K})$, $0 \leq t \leq ML$. Denote by $q_M(t)$ the uniquely determined M-th degree polynomial in t satisfying

$$q_M(t) = f(t) \qquad , t = 0, L, \ldots, ML.$$

Using Lagrange interpolation we can write

$$q_M(t) = \sum_{m=0}^{M} h_m(t) \exp(-2\pi i \frac{mn}{R})$$

where $h_0(t), \ldots, h_M(t)$ are polynomials in t. Choose a real number $d_M(n)$ such that

$$p_M(t) = d_M(n)\ q_M(t)$$

is the best-approximation to $f(t)$, in the sup-norm sense, over the interv-l $0 \leq t \leq ML$, within the set of real multiples of $q_M(t)$. We call $p_M(t)$ an M-th degree approximation since the interpolating polynomial has degree M.

Substituting $p_M(\ell)$ for $\exp(-2\pi i \frac{n\ell}{K})$, in (4.2), yields

(4.12) $$d_M(n)\ C_n(j)$$

where

(4.13) $$c_r(j) = \sum_{\ell=0}^{L-1} \sum_{m=0}^{M} h_m(\ell)\ u_{(r-m)L+\ell}(j)$$

and $C(j)$ is the DFT(R) of $c(j)$.

The analogy with section 3 can be seen by defining $h(k)$, $0 \leq k < K$, by

$$h(k) = h_{n-m}(\ell - m\,L) \quad L_m \leq \ell < L(m+1) \quad , 0 \leq m \leq M$$

and by considering the (M+1) L-tap filter

$$w_t(j) = \sum_{\ell=0}^{(M+1)L-1} h(\ell)\, u_{t+\ell-ML}(j).$$

Then, $c(j)$ is the (L:1) decimation of this filter; namely,

$$c_r(j) = w_{rL}(j).$$

Arguing as in the previous case, we can express the sequence $(c_r(0), \ldots, c_r(J))$ as the first (J+1)-outputs of an $(M+1) \cdot L$-tap filter. The number of complex multiplications used to compute $d_M(n)\, C_n(j)$, $-N \leq n \leq N$, $0 \leq j \leq J$, by this method is

$$(4.15) \qquad (M+1)K + R(J + (M+1)L + 3FFT(J + (M+1)L)) + (J+1)(FFT(R) + 2N + 1).$$

References

[1] J.W. Cooley and S. Winograd, A Limited Range Discrete Fourier Transform Algorithm, IEEE, International Conference on Acoustics, Speech and Signal Processing, April 1980, pp. 213-217.